STEP 1: IMPORTING DATA

In [71]:
```
import pandas as pd

ratings = pd.read_csv("ratings.csv")
movies  = pd.read_csv("movies.csv")
tags    = pd.read_csv("tags.csv")
links   = pd.read_csv("links.csv")
```

In [72]:
```
### CHECKING IMPORTED DATA
```

In [73]:
```
movies.head(), ratings.head(), tags.head(), links.head()
```

Out[73]:
```
(   movieId                               title  \
 0        1                    Toy Story (1995)
 1        2                      Jumanji (1995)
 2        3             Grumpier Old Men (1995)
 3        4            Waiting to Exhale (1995)
 4        5  Father of the Bride Part II (1995)

                                        genres
 0  Adventure|Animation|Children|Comedy|Fantasy
 1                   Adventure|Children|Fantasy
 2                               Comedy|Romance
 3                         Comedy|Drama|Romance
 4                                       Comedy  ,
    userId  movieId  rating  timestamp
 0       1        1     4.0  964982703
 1       1        3     4.0  964981247
 2       1        6     4.0  964982224
 3       1       47     5.0  964983815
 4       1       50     5.0  964982931,
    userId  movieId              tag   timestamp
 0       2    60756            funny  1445714994
 1       2    60756  Highly quotable  1445714996
 2       2    60756     will ferrell  1445714992
 3       2    89774     Boxing story  1445715207
 4       2    89774              MMA  1445715200,
    movieId  imdbId   tmdbId
 0        1  114709    862.0
 1        2  113497   8844.0
 2        3  113228  15602.0
 3        4  114885  31357.0
 4        5  113041  11862.0)
```

STEP 2: CHECK FOR DUPLICATE AND MISSING VALUES

In [69]:
```
#Drop duplicate
movies = movies.drop_duplicates().copy()
tags = tags.drop_duplicates().copy()
links = links.drop_duplicates().copy()
```

In [70]:
```
ratings.isnull()
movies.isnull()
tags.isnull()
links.isnull()
```

Out[70]:

	movieId	imdbId	tmdbId
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False
...	...	...	...
9737	False	False	False
9738	False	False	False
9739	False	False	False
9740	False	False	False
9741	False	False	False

9742 rows × 3 columns

STEP 3: CONVERT TIMESTAMP

In [19]:
```
ratings['timestamp'] = pd.to_datetime(ratings['timestamp'], unit='s')
```

In [20]:
```
### CHECK RATINGS AND MOVIES INDEX
```

In [21]:
```
print(movies.columns)
print(ratings.columns)
```

```
Index(['movieId', 'title', 'genres'], dtype='object')
Index(['userId', 'movieId', 'rating', 'timestamp'], dtype='object')
```

STEP 4: MERGE RATINGS AND MOVIES USING MOVIEID

In [22]:
```
df = ratings.merge(movies, on="movieId")
```

In [23]:
```
df.head()
```

Out[23]:

	userId	movieId	rating	timestamp	title	genre
0	1	1	4.0	2000-07-30 18:45:03	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantas
1	1	3	4.0	2000-07-30 18:20:47	Grumpier Old Men (1995)	Comedy|Romanc
2	1	6	4.0	2000-07-30 18:37:04	Heat (1995)	Action|Crime|Thrille
3	1	47	5.0	2000-07-30 19:03:35	Seven (a.k.a. Se7en) (1995)	Mystery|Thrille
4	1	50	5.0	2000-07-30 18:48:51	Usual Suspects, The (1995)	Crime|Mystery|Thrille

STEP 5: FEATURE ENGINEERING

In [24]:
```
### IMPORT LIBRARIES NEED
```

In [25]:
```
import re
import numpy as np
```

1. EXTRACT YEAR FROM MOVIE TITLE AND CONVERT YEAR TO INTEGER

In [26]:
```
def extract_year_from_title(title):
    m = re.search(r'\((\d{4})\)$', str(title))
    if m:
        year = int(m.group(1))
        if 1874 <= year <= pd.Timestamp.now().year + 1:  # sanity check
            return year
```

In [27]:
```
### REMOVE YEAR FROM TITLE
```

In [28]:
```
def strip_year_from_title(title):
    return re.sub(r'\s*\(\d{4}\)\s*$', '', str(title)).strip()
```

In [29]:
```
### CREATE COLUMNS FOR EXTRACTED YEAR AND TITLE WITHOUT YEAR
```

In [30]:
```
df['extracted_year'] = df['title'].apply(extract_year_from_title).astype('Int64')
df['title_without_year'] = df['title'].apply(strip_year_from_title)
```

In [31]:
```
### VIEW NEW COLUMNS(EXTRACTED_YEAR AND TITLE_WITHOUT_YEAR
```

In [32]:
```
df[['title','extracted_year','title_without_year']].head()
```

Out[32]:

	title	extracted_year	title_without_year
0	Toy Story (1995)	1995	Toy Story
1	Grumpier Old Men (1995)	1995	Grumpier Old Men
2	Heat (1995)	1995	Heat
3	Seven (a.k.a. Se7en) (1995)	1995	Seven (a.k.a. Se7en)
4	Usual Suspects, The (1995)	1995	Usual Suspects, The

In [33]:
```
### DEFINE THE GENRES, SPLIT WHERE '|'
```

In [34]:
```
def split_genres(g):
    return str(g).split('|')
```

2. COUNT AND CREATE NUMBER OF GENRES AND MULTI GENRE COLUMN

In [35]:
```
df['number_genres'] = df['genres'].apply(lambda g: len(split_genres(g))).astype('In
df['multi_genre'] = (df['number_genres'] > 1).astype(int)
```

In [36]:
```
### VIEW NEW COLUMN ( NUMBER_GENRES AND MULTI_GENRE)
```

In [37]:
```
df[['genres','number_genres','multi_genre']].head()
```

Out[37]:

	genres	number_genres	multi_genre
0	Adventure|Animation|Children|Comedy|Fantasy	5	1
1	Comedy|Romance	2	1
2	Action|Crime|Thriller	3	1
3	Mystery|Thriller	2	1
4	Crime|Mystery|Thriller	3	1

3. EXTRACT YEAR, MONTH, DAY OF THE WEEK AND HOUR FROM TIMESTAMP, NEW COLUMNS

In [38]:
```
df['rating_year']  = df['timestamp'].dt.year
df['rating_month'] = df['timestamp'].dt.month
```

```
df['rating_dow']   = df['timestamp'].dt.dayofweek   # 0=Monday
df['rating_hour']  = df['timestamp'].dt.hour
```

In [39]:
```
### CHECK NEW COLUMNS ( RATING_YEAR, RATING_MONTH, RATING_DOW AND RATING_HOUR)
```

In [40]:
```
df[['timestamp','rating_year','rating_month','rating_dow','rating_hour']].head()
```

Out[40]:

	timestamp	rating_year	rating_month	rating_dow	rating_hour
0	2000-07-30 18:45:03	2000	7	6	18
1	2000-07-30 18:20:47	2000	7	6	18
2	2000-07-30 18:37:04	2000	7	6	18
3	2000-07-30 19:03:35	2000	7	6	19
4	2000-07-30 18:48:51	2000	7	6	18

In [ ]:
```

```

4. COUNT HOW MANY TAGS EACH MOVIE HAS AND CREATED A COLUMN

In [45]:
```
tag_counts = tags.groupby('movieId', as_index=False)['tag'].count().rename(columns=
```

In [46]:
```
### MERGE WITH MOVIE DATA USING MOVIEID
```

In [47]:
```
df = df.merge(tag_counts, on='movieId', how='left')
```

In [48]:
```
### REPLACE MISSING VALUES AND CONVERT TO INTEGERS
```

In [49]:
```
df['tag_count'] = df['tag_count'].fillna(0).astype(int)
```

In [50]:
```
### VIEW COLUMN
```

In [51]:
```
df[['movieId','title','tag_count']].head()
```

Out[51]:

	movieId	title	tag_count
0	1	Toy Story (1995)	3
1	3	Grumpier Old Men (1995)	2
2	6	Heat (1995)	0
3	47	Seven (a.k.a. Se7en) (1995)	3
4	50	Usual Suspects, The (1995)	6

5. TOP 10 MOST RATED MOVIE

In [52]:
```
top_movies = (
    df.groupby('title_without_year')['rating']
      .count()
      .sort_values(ascending=False)
      .head(10)
)

top_movies
```

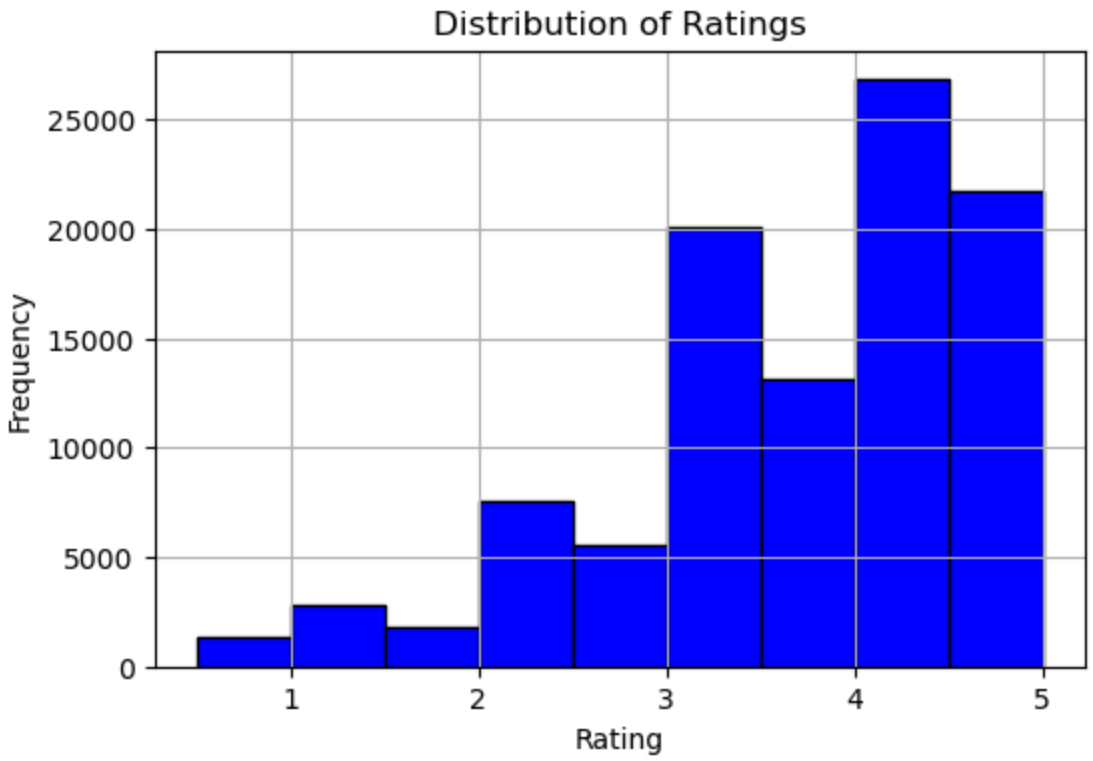
Out[52]:
```
title_without_year
Forrest Gump                            329
Shawshank Redemption, The               317
Pulp Fiction                            307
Silence of the Lambs, The               279
Matrix, The                             278
Star Wars: Episode IV - A New Hope      251
Jurassic Park                           238
Braveheart                              237
Terminator 2: Judgment Day              224
Schindler's List                        220
Name: rating, dtype: int64
```

In [ ]:
```
```

In [83]:
```
# Finalize types
num_cols = ['number_genres','multi_genre','rating_year','rating_month','rating_dow'
for c in num_cols:
    df[c] = pd.to_numeric(df[c], errors='coerce')
```

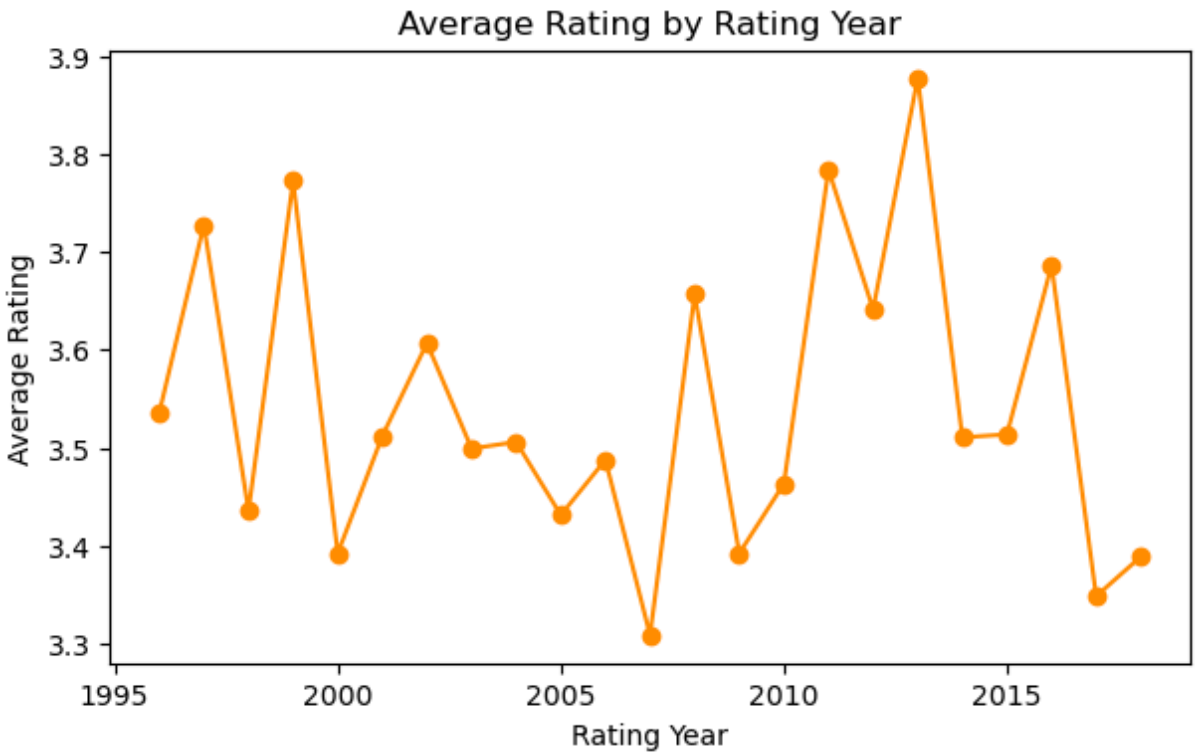
STEP 6: EXPLORATORY DATA ANALYSIS

In [84]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(6,4))
df['rating'].hist(bins=[0.5,1,1.5,2,2.5,3,3.5,4,4.5,5,], edgecolor='black',color='b
plt.title("Distribution of Ratings")
plt.xlabel("Rating")
plt.ylabel("Frequency")
plt.show()
```



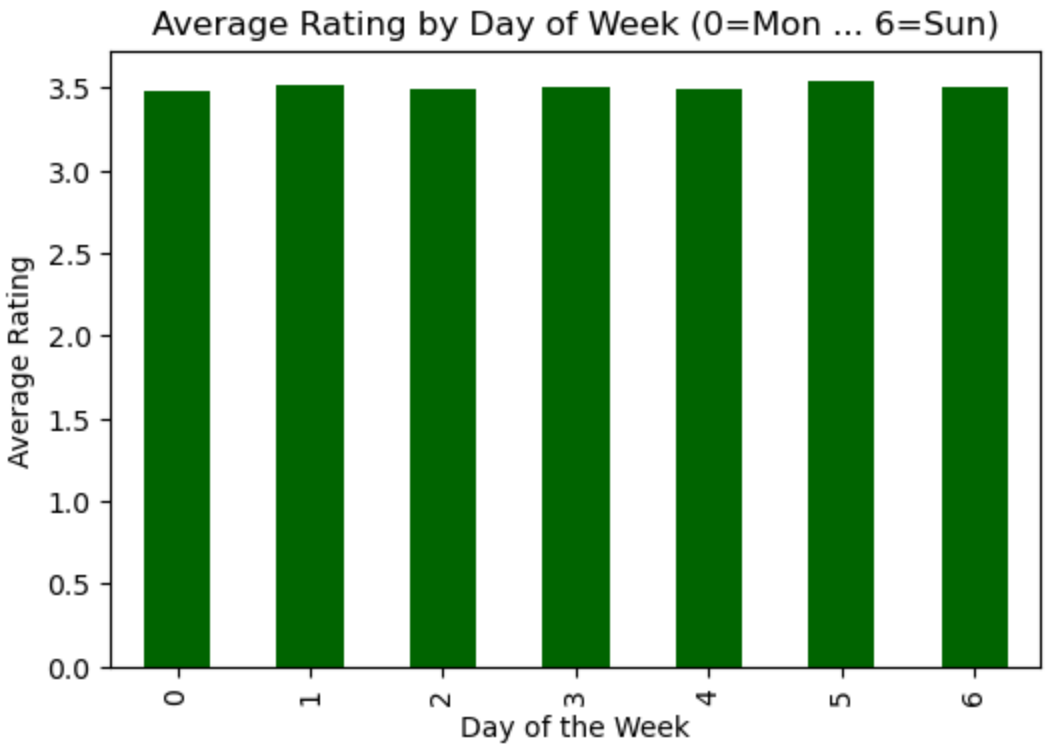
In [ ]:

In [85]:
```
year_avg = df.groupby('rating_year')['rating'].mean()

plt.figure(figsize=(7,4))
year_avg.plot(kind='line', marker='o', color='darkorange')
plt.title("Average Rating by Rating Year")
plt.xlabel("Rating Year")
plt.ylabel("Average Rating")
plt.show()

year_avg.head()
```



Out[85]:
```
rating_year
1996    3.535927
1997    3.727557
1998    3.435897
1999    3.772448
2000    3.392506
Name: rating, dtype: float64
```

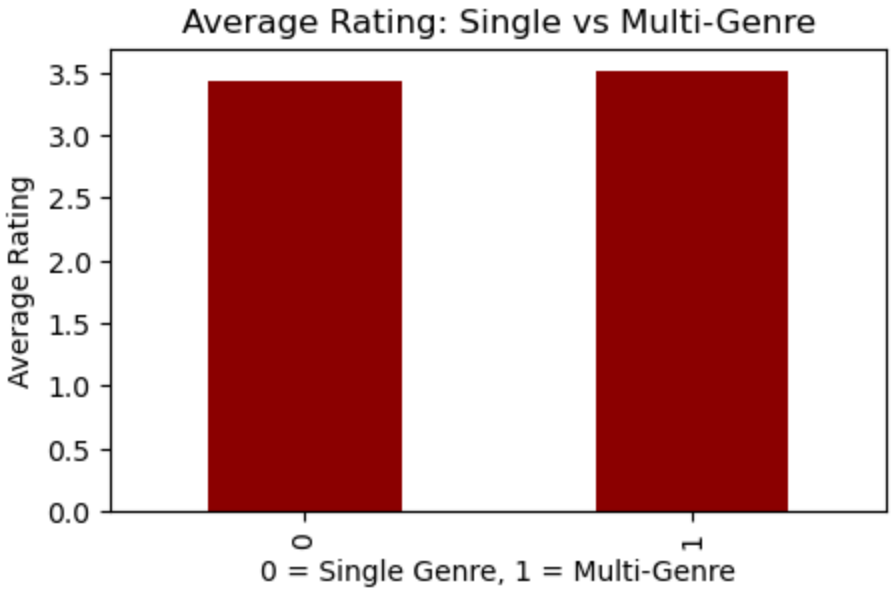
In [86]:
```
dow_avg = df.groupby('rating_dow')['rating'].mean()

plt.figure(figsize=(6,4))
dow_avg.plot(kind='bar', color='darkgreen')
plt.title("Average Rating by Day of Week (0=Mon ... 6=Sun)")
plt.xlabel("Day of the Week")
plt.ylabel("Average Rating")
plt.show()

dow_avg
```



Out[86]:
```
rating_dow
0    3.480493
1    3.518159
2    3.484872
3    3.497104
4    3.494604
5    3.543987
6    3.501172
Name: rating, dtype: float64
```

In [ ]:
```
```

In [87]:
```
multi_vs_single = df.groupby('multi_genre')['rating'].mean()

plt.figure(figsize=(5,3))
multi_vs_single.plot(kind='bar', color='darkred')
plt.title("Average Rating: Single vs Multi-Genre")
plt.xlabel("0 = Single Genre, 1 = Multi-Genre")
plt.ylabel("Average Rating")
plt.show()

multi_vs_single
```

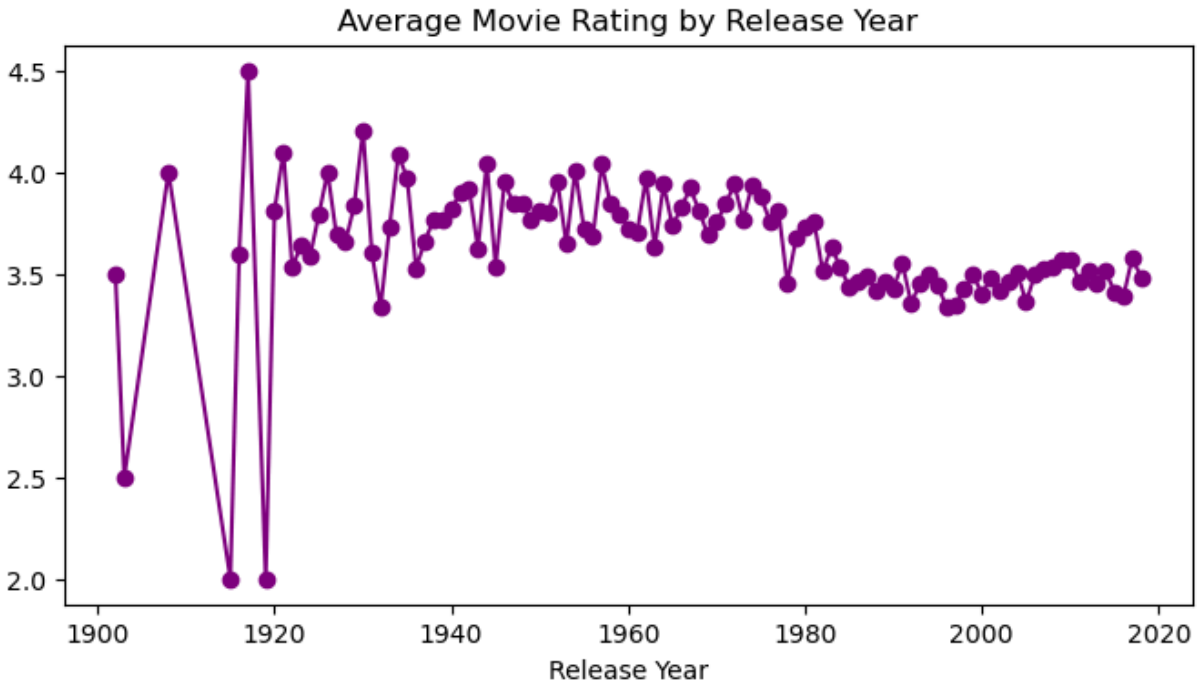


Out[87]:
```
multi_genre
0    3.429846
1    3.515500
Name: rating, dtype: float64
```

In [ ]:

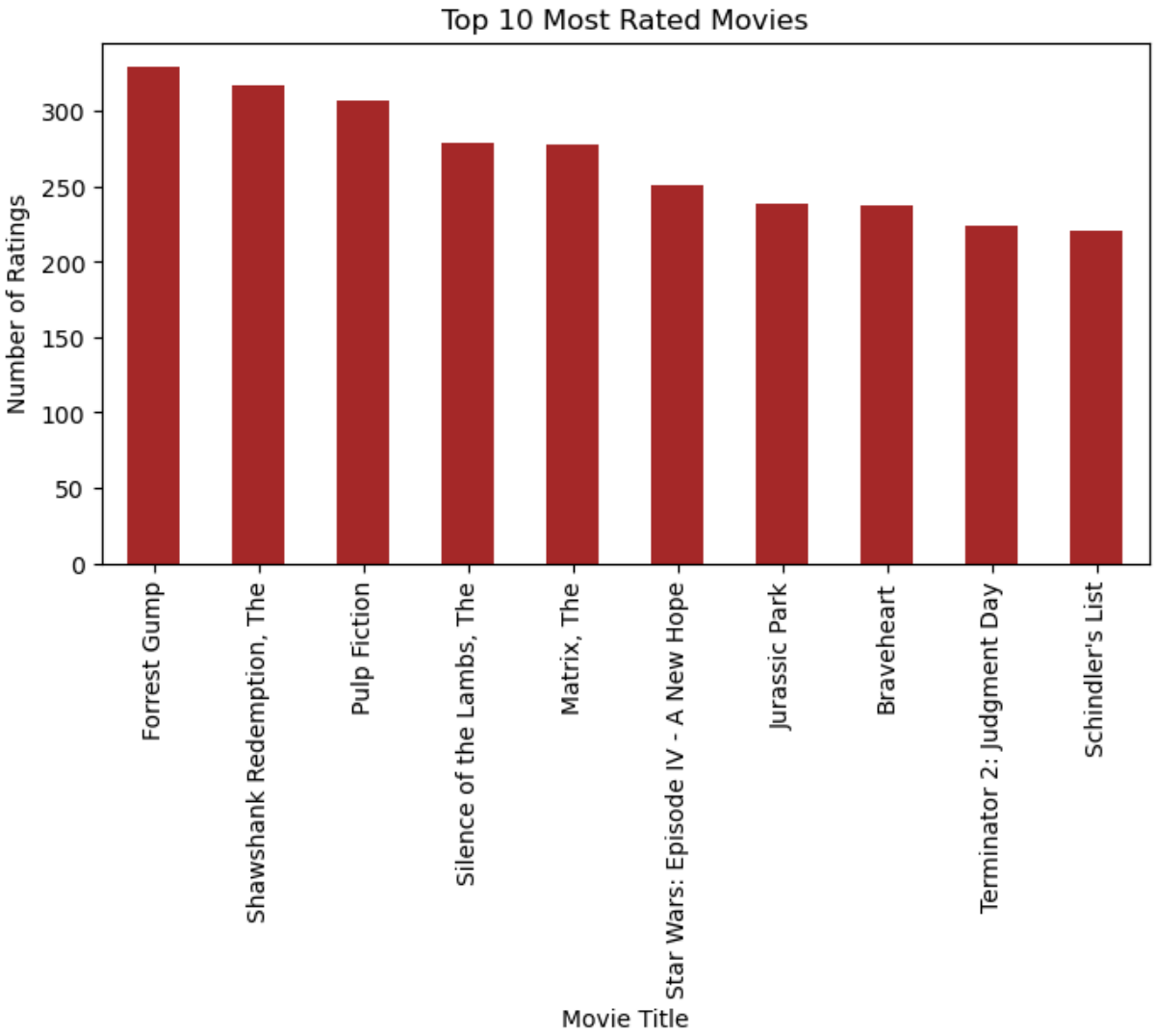
In [88]:
```
release_year_avg = df.groupby('extracted_year')['rating'].mean().dropna()

plt.figure(figsize=(8,4))
release_year_avg.plot(kind='line', marker='o', color='purple')
plt.title("Average Movie Rating by Release Year")
plt.xlabel("Release Year")
```

Out[88]:
```
Text(0.5, 0, 'Release Year')
```



In [ ]:

In [89]:
```
plt.figure(figsize=(8,4))
top_movies.plot(kind='bar', color='brown')
plt.title("Top 10 Most Rated Movies")
plt.xlabel("Movie Title")
plt.ylabel("Number of Ratings")
plt.show()
```



STEP 7: EXPORT CLEANED DATA

In [91]:
```
df.to_csv('cleaned_movielens_with_features.csv', index=False)
```

In [ ]: